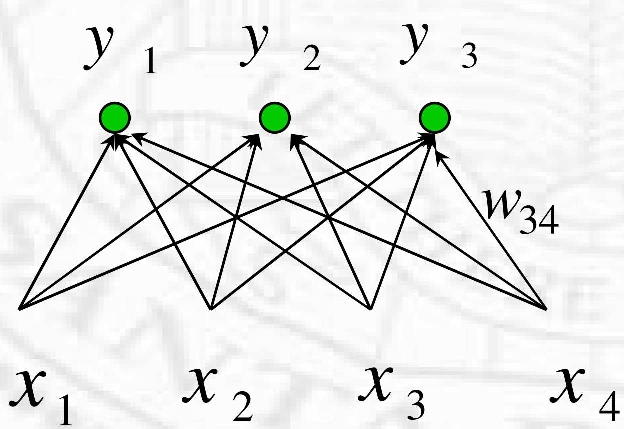
Simple Perceptrons

Simple Perceptrons

- Perform supervised learning
  - correct I/O associations are provided
- Feed-forward networks
  - connections are one-directional
- One layer
  - input layer + output layer

Notations

- *N*: dimension of the input vector
- *M*: dimension of the output vector
- inputs $x_j, j = 1, \ldots, N$
- real outputs $y_i, i = 1, \ldots, M$
- weight vectors $w_{ij}, i = 1, \ldots, M, \; j = 1, \ldots, N$
- activation function g



$$y_i = g(net_i) = g\left(\sum_{j=1}^{N} w_{ij} x_j\right)$$

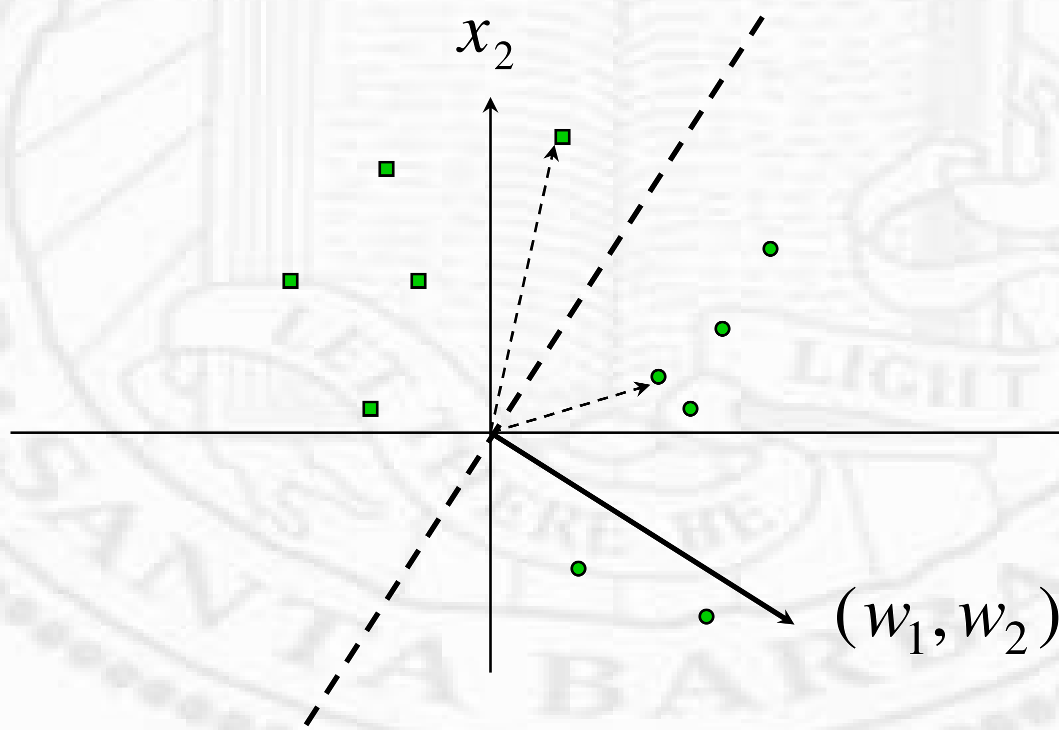
Perceptron Training

- given
  - input patterns $\mathbf{x}^u$
  - desired output patterns $\mathbf{O}^u$
  - how to adapt the connection weights such that the actual outputs conform the desired outputs

$$O_i^u = y_i^u \quad i = 1, \cdots, M$$

Simplest case

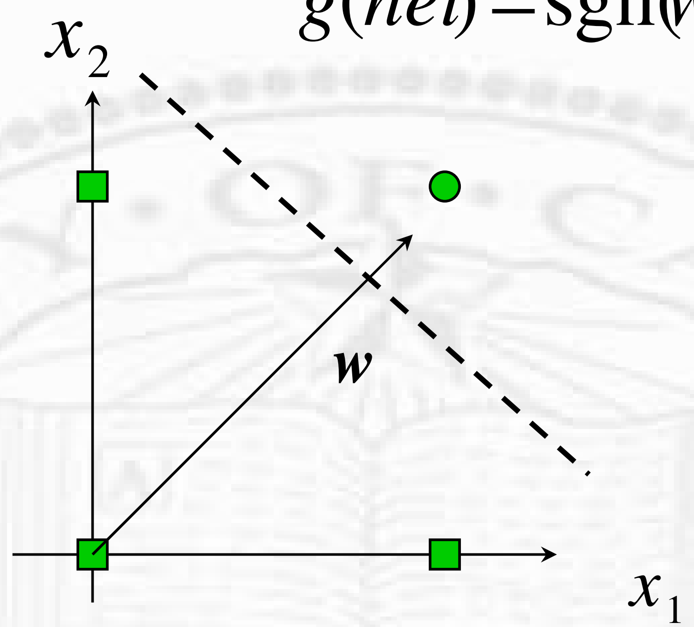
- two types of inputs
- binary outputs (-1,1)
- thresholding $\quad \mathrm{sgn}(\mathbf{w}\cdot\mathbf{x}) = \mathrm{sgn}(w_1 x_1{}^u + w_2 x_2{}^u + w_o) = \mathbf{y}^u$

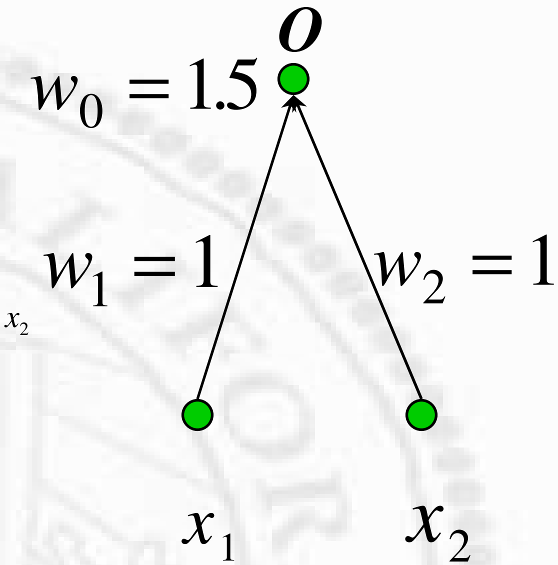


Examples

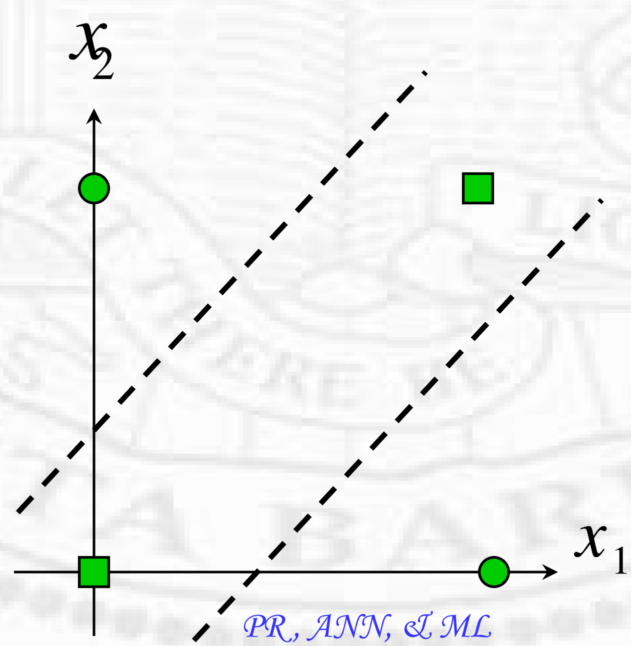
$$g(net) = \mathrm{sgn}(w_1 x_1^u + w_2 x_2^u - 1.5)$$

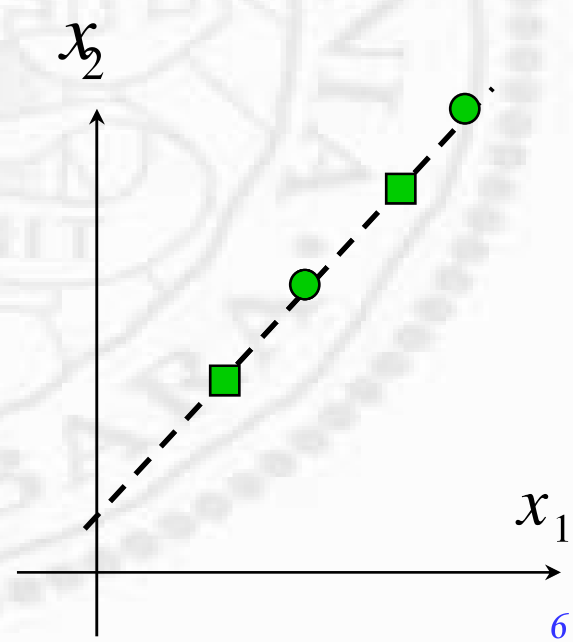
x_1	x_2	O
0	0	-1
0	1	-1
1	0	-1
1	1	1





x_1	x_2	O
0	0	-1
0	1	1
1	0	1
1	1	-1





Linear separability

- Is it at all possible to learn the desired I/O associations?
  - yes, if w_{ij} can be found such that

$$O_i^u = \text{sgn}(\sum_{j=1}^{N} w_{ij} x_j^u - w_{i0}) = y_i^u \text{ for all i and u}$$

  - no, otherwise
- Single-layer perceptron is severely limited in what it can learn

Perceptron Learning

- Linear separable or not, how to find the set of weights?
- Using tagged samples
  - closed form solution
  - iterative solutions

Closed Form Solution

$$\begin{bmatrix} x_1^1 & \cdots & x_n^1 & 1 \\ x_1^2 & \cdots & x_n^2 & 1 \\ \ddots & \ddots & \ddots & \ddots \\ x_1^u & \cdots & x_n^u & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_o \end{bmatrix} = \begin{bmatrix} O^1 \\ O^2 \\ \vdots \\ O^u \end{bmatrix}$$

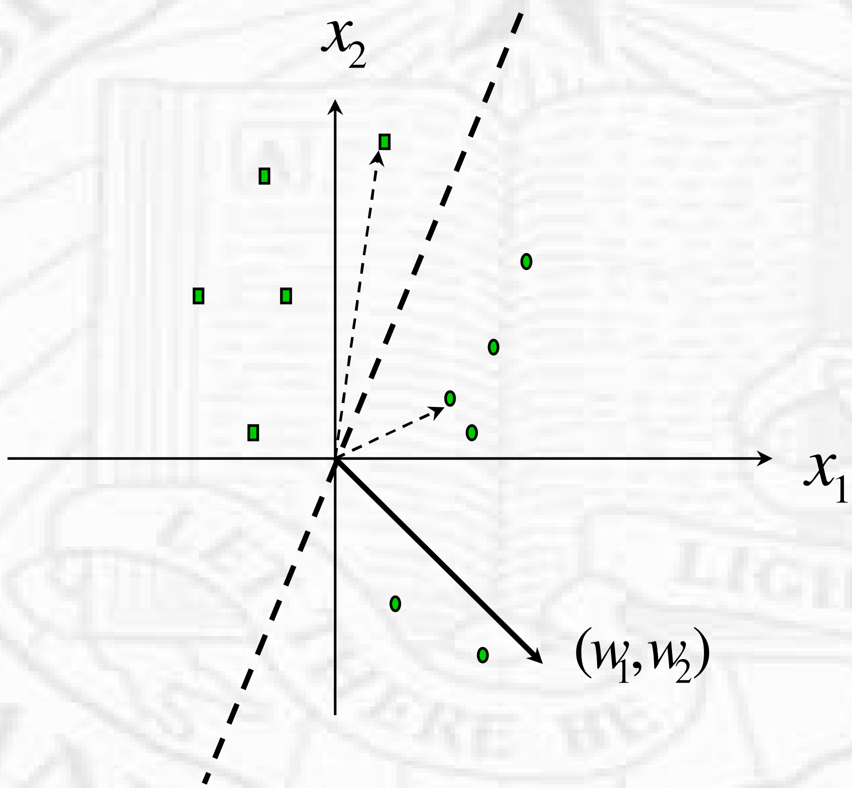
$$\mathbf{AW} = \mathbf{B}$$

$$\mathbf{W} = (\mathbf{A}^{\mathbf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{B}$$

- Not practical when number of samples is large (most likely case)

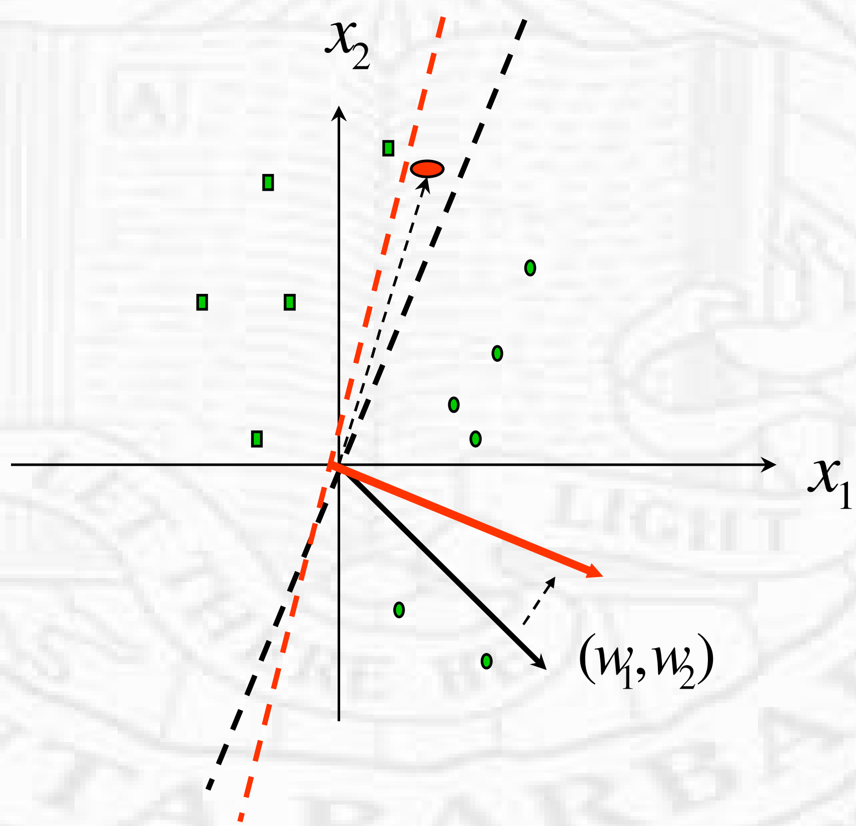
Perceptron Learning Rule

- If a pattern is correctly classified, no action



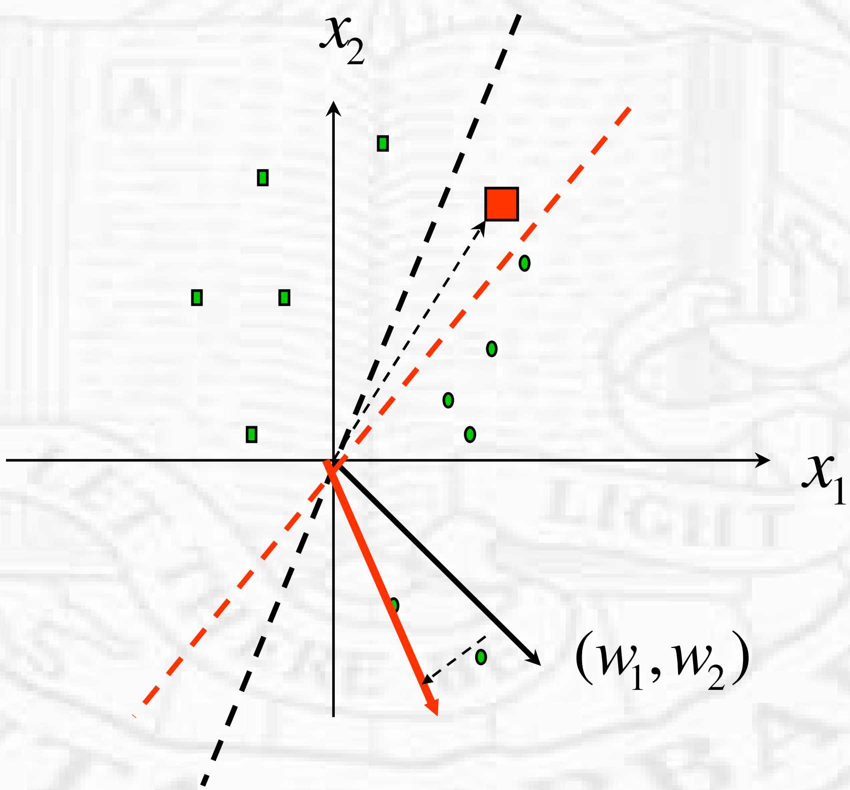
Perceptron Learning Rule (cont.)

- If a positive pattern becomes a negative pattern



Perceptron Learning Rule (cont.)

- If a negative pattern becomes a positive pattern



$$\mathbf{w}^{(k+1)} = \begin{cases} \mathbf{w}^{(k)} + c\mathbf{x} & \mathbf{w}^{(k)} \cdot \mathbf{x} < 0, \mathbf{x} \in + \\ \mathbf{w}^{(k)} - c\mathbf{x} & \mathbf{w}^{(k)} \cdot \mathbf{x} > 0, \mathbf{x} \in - \\ \mathbf{w}^{(k)} & otherwise \end{cases}$$

$$\mathbf{w}^{(k+1)} = \begin{cases} \mathbf{w}^{(k)} + cy\mathbf{x} & y(\mathbf{w}^{(k)} \cdot \mathbf{x}) < 0, \mathbf{x} \in + or - \\ \mathbf{w}^{(k)} & otherwise \end{cases}$$

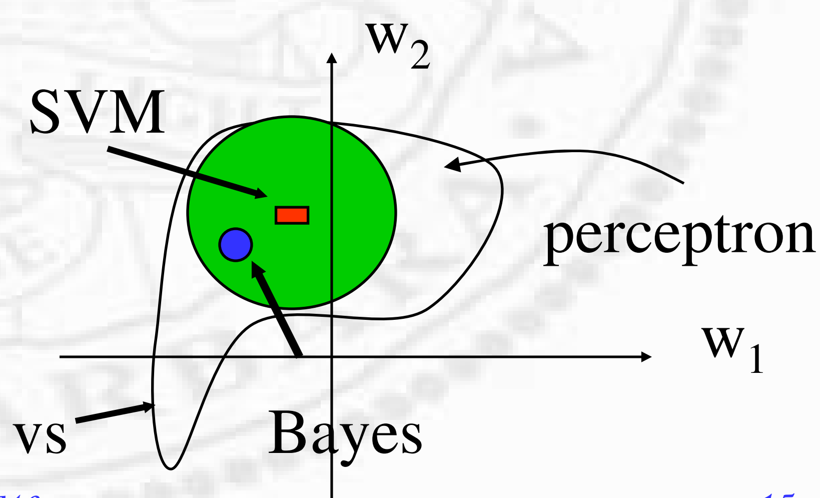
- How should c be decided?
  - Fixed increment
  - Fractional correction

Perceptron Learning Rule (cont.)

- Weight is a signed, linear combination of training points
- Use those informative points (those the classifier made a mistake, mistake driven)
- This is VERY important, lead later to generalization to Support Vector Machines

Comparison

- Version space
  - The (w1-w2) space of all feasible solutions
- Perceptron learning
  - Greedy, gradient descent that often ends up at boundary of the version space with little space for error
- SVM learning
  - Center of largest imbedded sphere in the version space (maximum margin)
- Bayes point machine
  - Centroid of the version space



Perceptron Usage Rules

- After the weight has been determined

$$y = \mathbf{w} \cdot \mathbf{x} = (\sum_i \alpha_i y_i \mathbf{x}_i) \cdot \mathbf{x} = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}$$

- Classification involves inner product of training samples and test samples
- This is again VERY important, lead later to generalization to Kernel Methods

Hebb's Learning Rule

- Synapse strength should be increased when both pre- and post-synapse neurons fire vigorously
  - for binary outputs

$$\mathbf{w}_{ij}^{new} = \mathbf{w}_{ij}^{old} + \Delta\mathbf{w}_{ij}$$

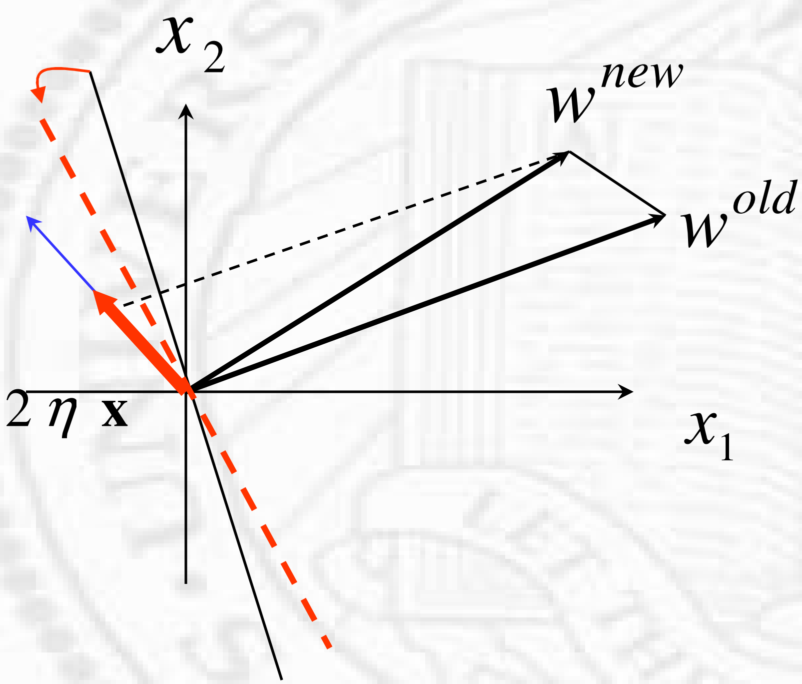
$$\Delta\mathbf{w}_{ij} = \begin{cases} 2\eta y_i^u x_j^u & \text{if } y_i^u \neq O_i^u \\ 0 & \text{otherwise} \end{cases}$$

$$= \eta(1 - y_i^u O_i^u)\, y_i^u x_j^u$$

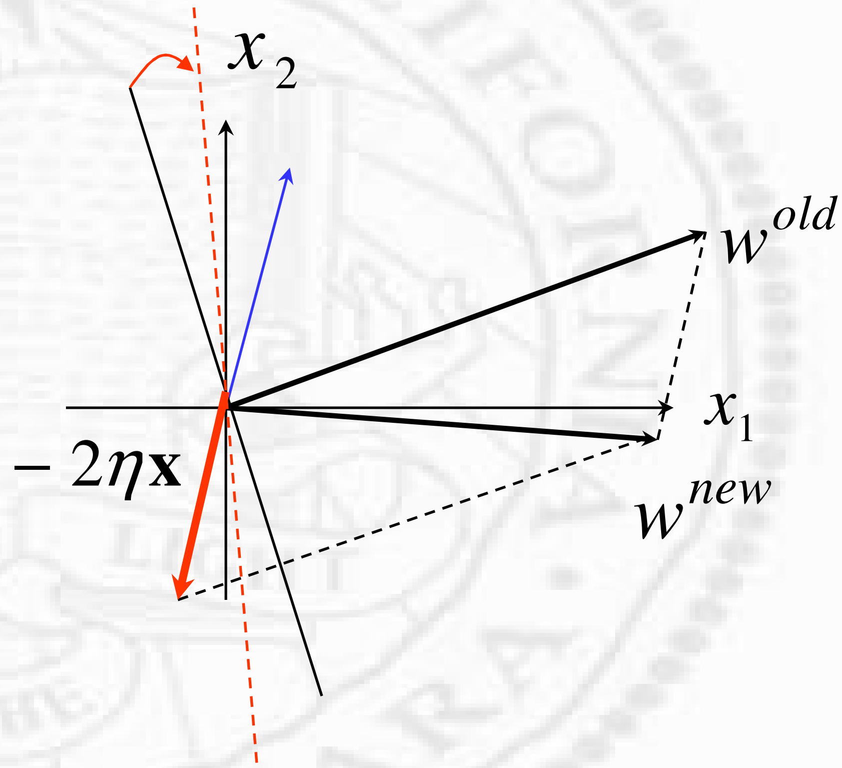
$$= \eta(y_i^u - y_i^{u\,2} O_i^u)\, x_j^u$$

$$= \eta\,\underbrace{\boxed{(y_i^u - O_i^u)}}_{\delta}\, x_j^u$$

- Case 1: $O=-1, y=1$



- Case 2: $O=1, y=-1$



LMS (Widrow-Hoff, Delta)

- Not restricted to binary outputs
  - Gradient search

$$E(\mathbf{w}) = \frac{1}{2}\sum_u \sum_i (O_i^u - y_i^u)^2 = \frac{1}{2}\sum_u \sum_i (O_i^u - g(\sum_{j=1}^{N} w_{ij} x_j^u))^2$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_{ij}} = -\sum_u (O_i^u - g(net_i^u))\, g'(net_i^u) x_j^u$$

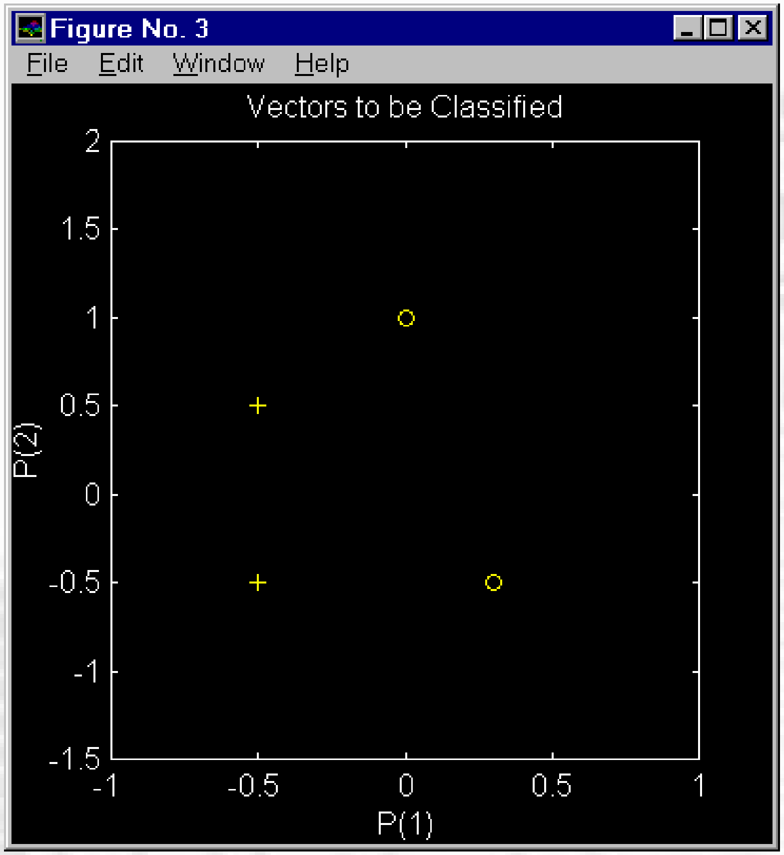
$$\mathbf{w}_{ij}^{new} = \mathbf{w}_{ij}^{old} + \Delta \mathbf{w}_{ij}$$

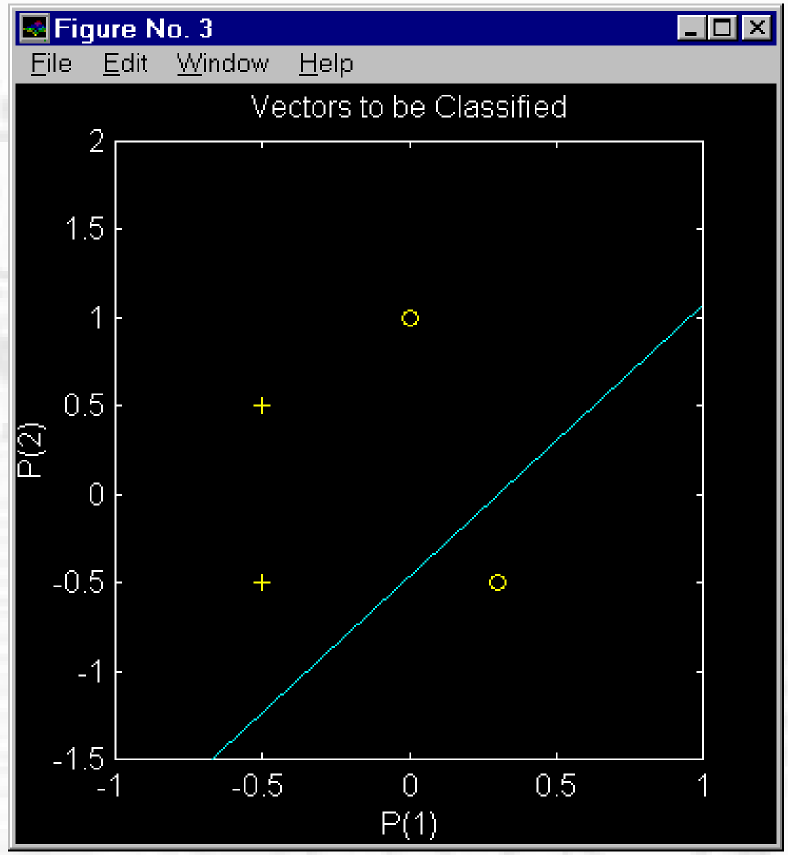
$$= \mathbf{w}_{ij}^{old} + \eta \sum_u (O_i^u - g(net_i^u))\, g'(net_i^u) x_j^u$$

Nothing but Chain Rule

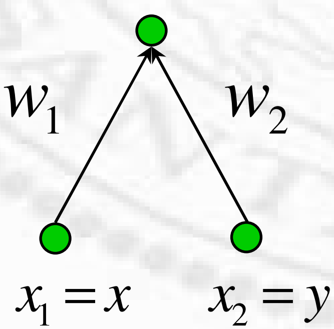
$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_{ij}} = \boxed{\frac{\partial (O_i^u - y_i^u)^2}{\partial (O_i^u - y_i^u)}} \boxed{\frac{\partial (O_i^u - y_i^u)}{\partial y_i^u}} \boxed{\frac{\partial y_i^u}{\partial net_i^u}} \boxed{\frac{\partial net_i^u}{\partial \mathbf{w}_{ij}}}$$

$$= \boxed{-} \sum_u \boxed{(O_i^u - g(net_i^u))} \boxed{g'(net_i^u)} \boxed{x_j^u}$$





$$O = g(net) = \mathrm{sgn}(w_1 x_1 + w_2 x_2 + b)$$

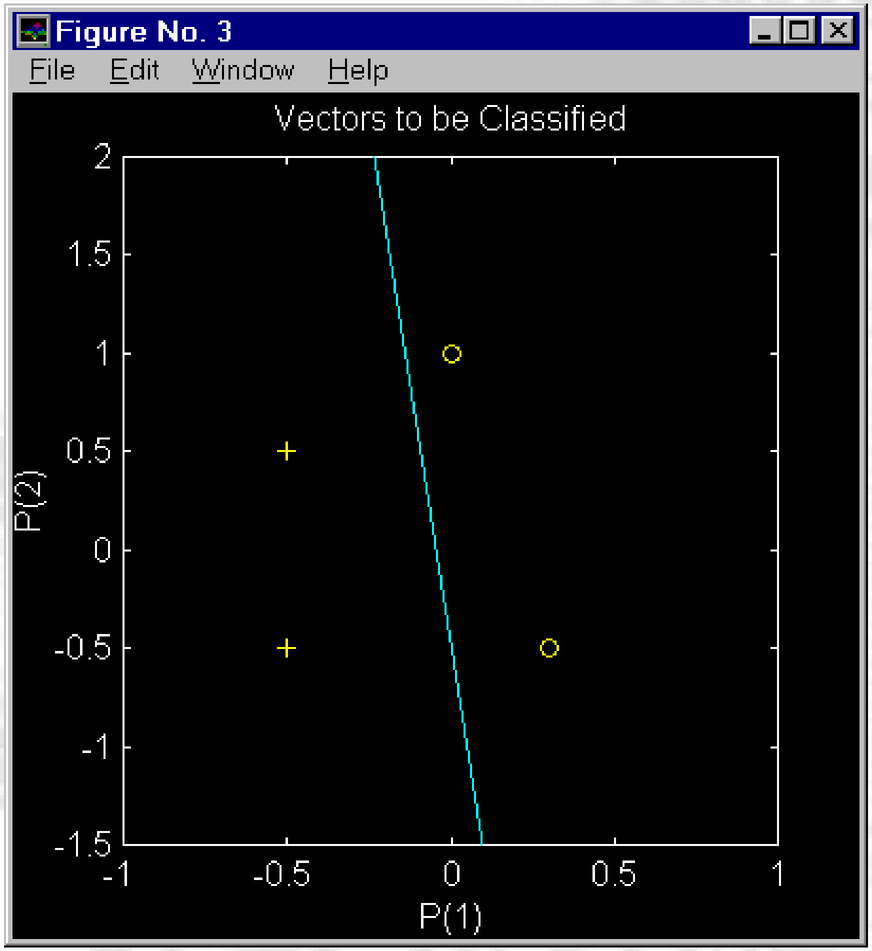


$$w_1 = 0.4299$$

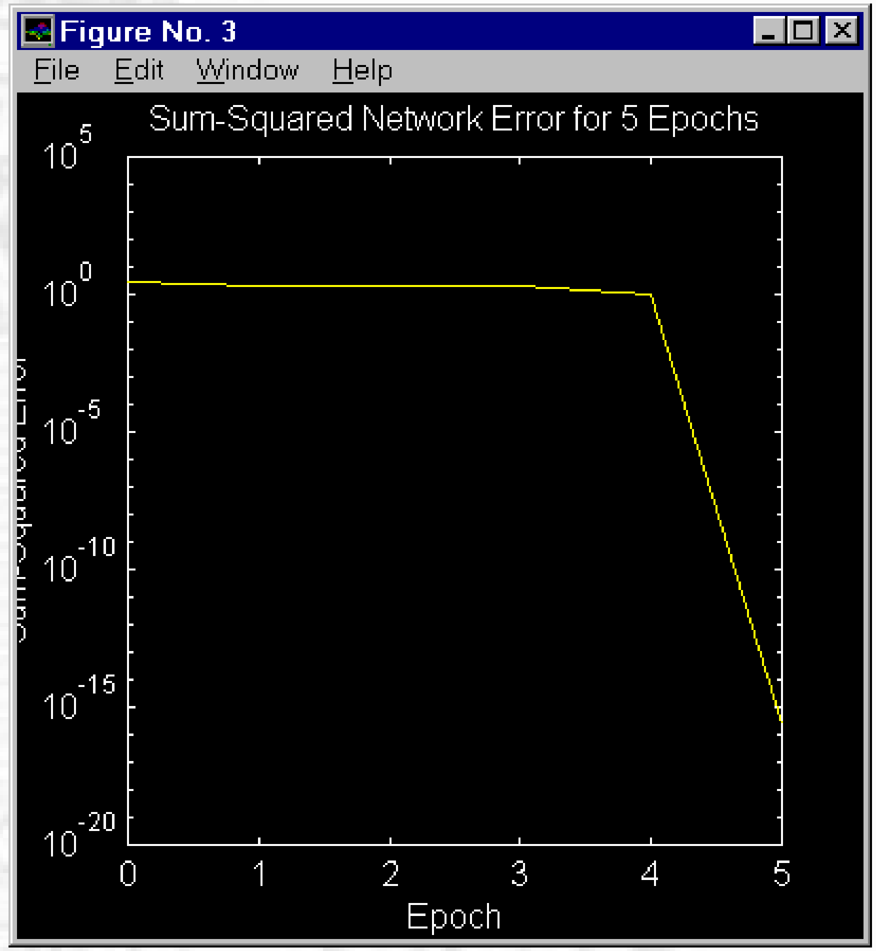
$$w_2 = -0.2793$$

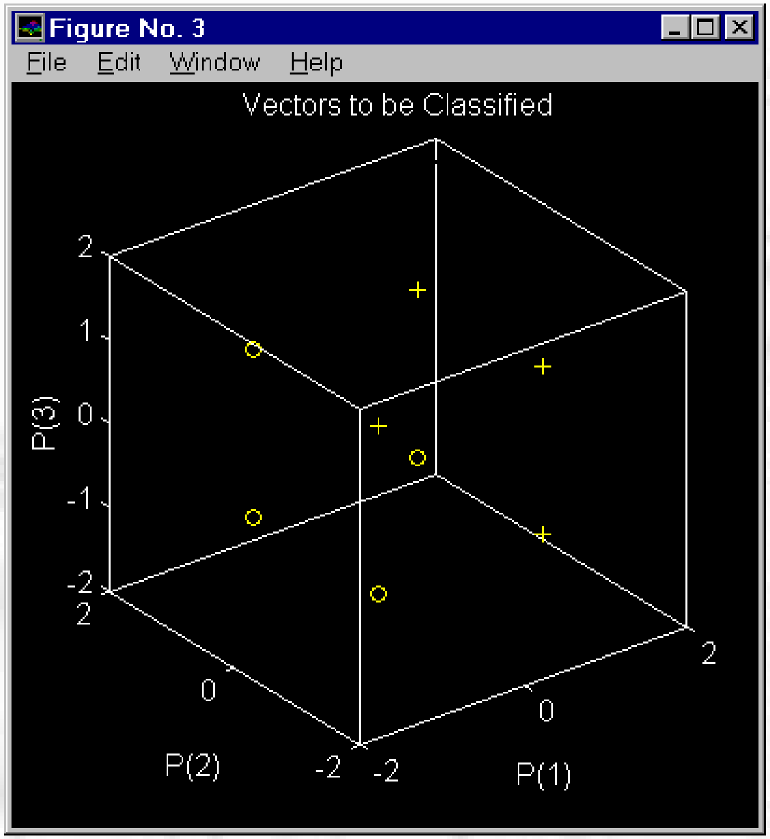
$$b = -0.1312$$

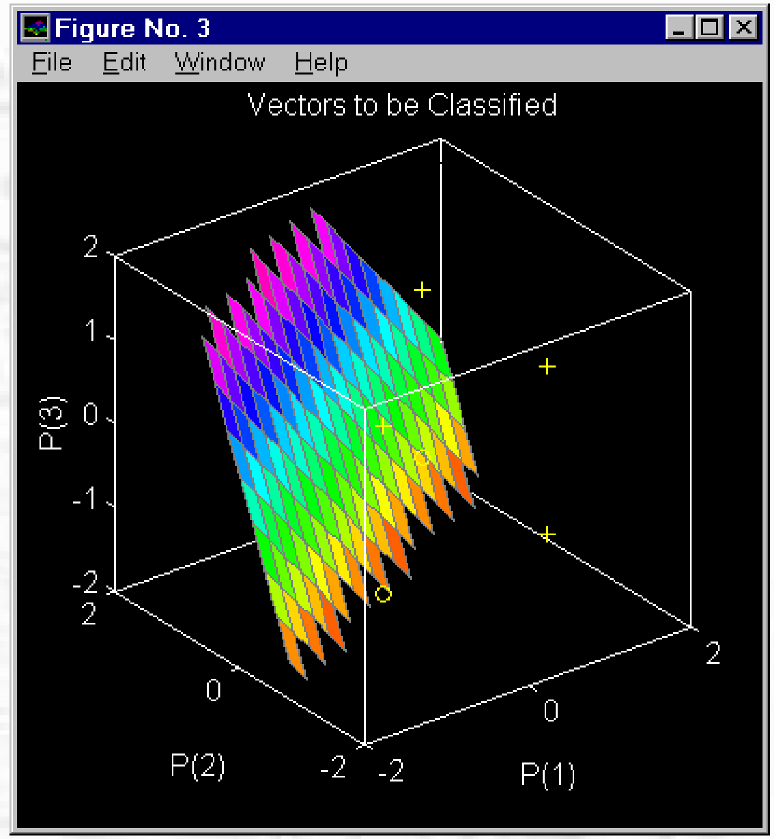
Final training results



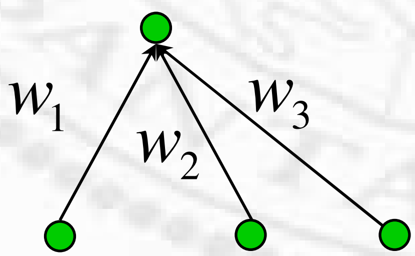
Error vs. training epoch







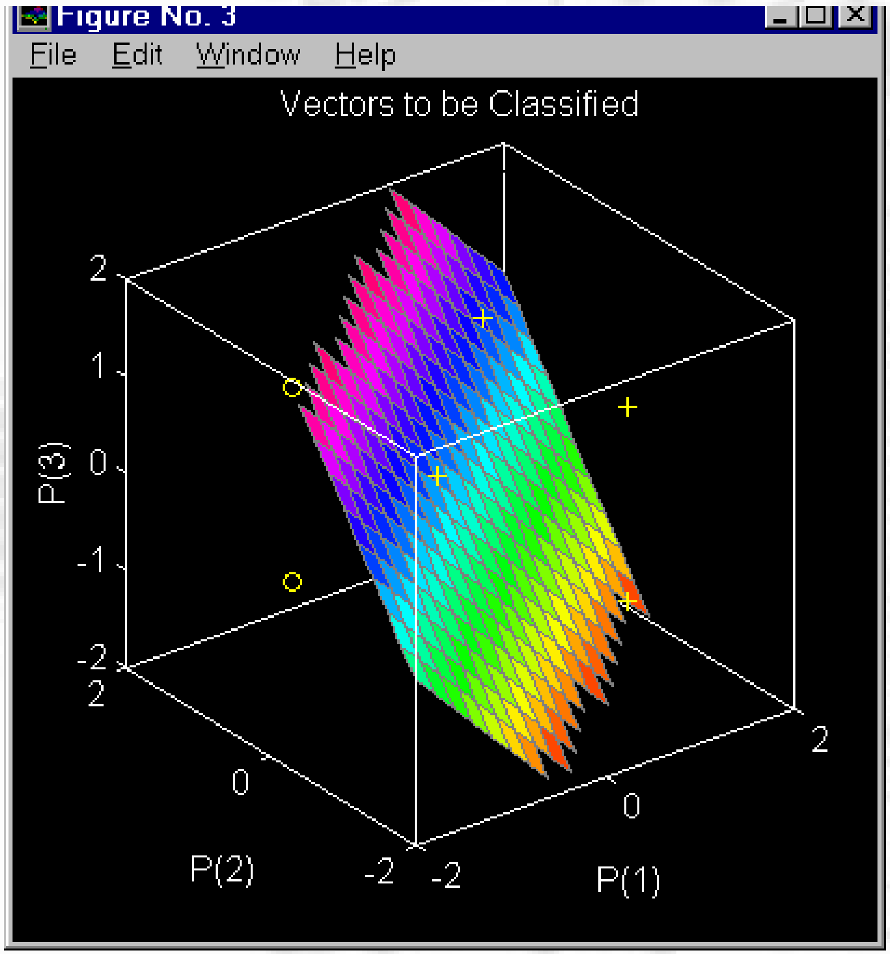
$$y = g(net) = \mathrm{sgn}(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$



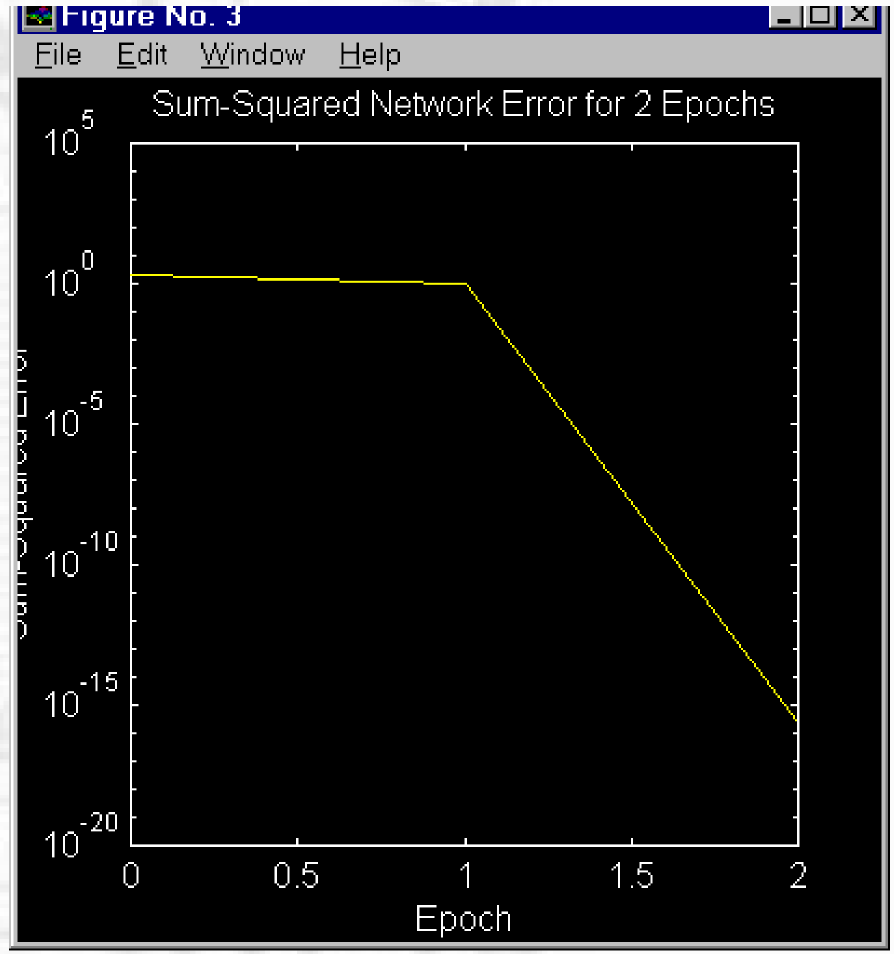
$x_1 = x$ $\quad x_2 = y$ $\quad x_3 = z$

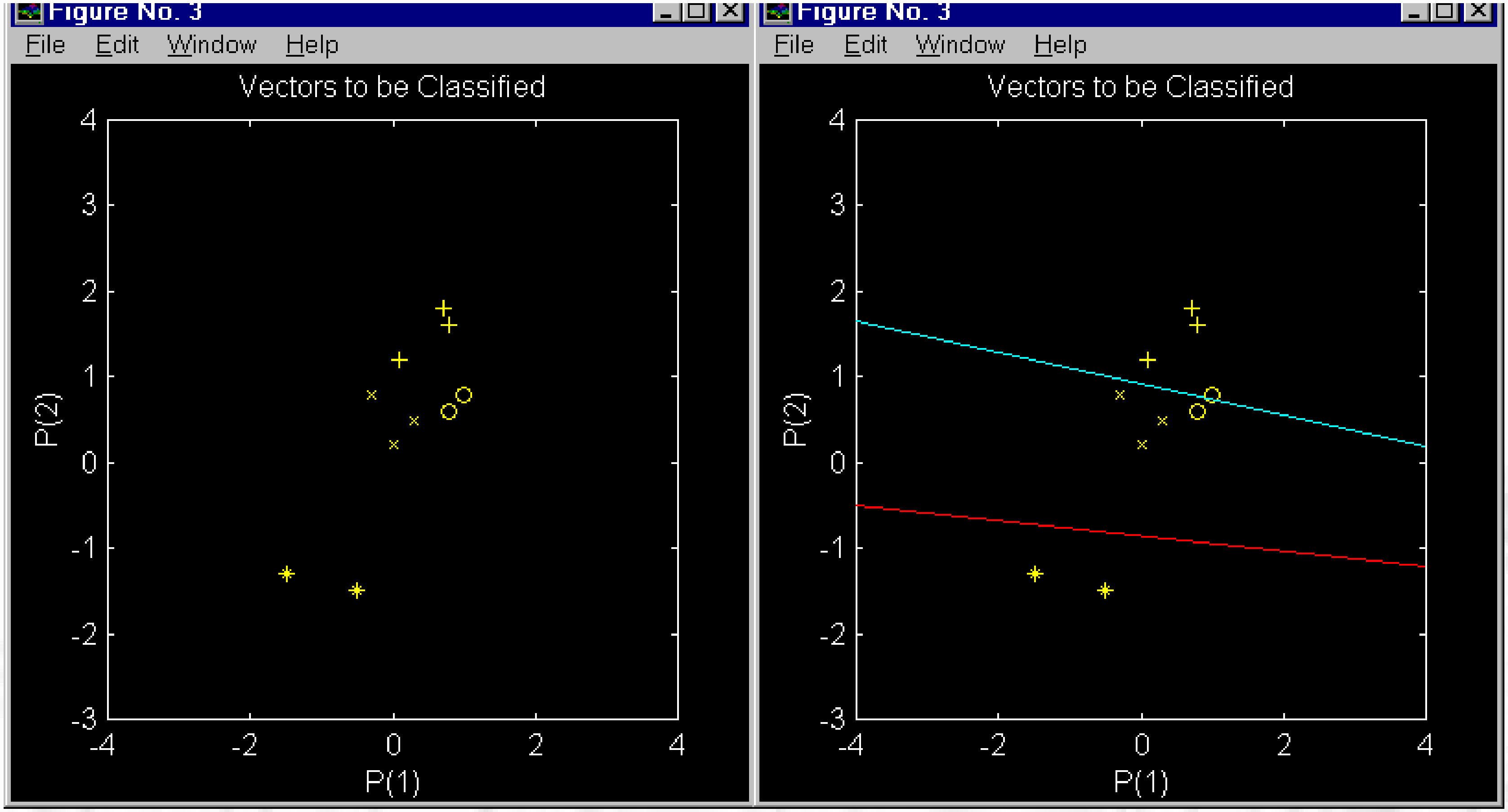
$$w_1 = 0.4232$$
$$w_2 = -0.7411$$
$$w_3 = -0.3196$$
$$b = 0.7550$$

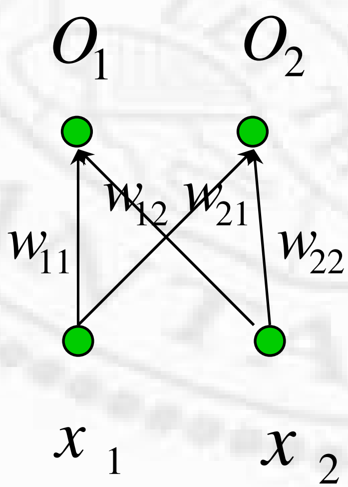
Final training results



Error vs. training epoch



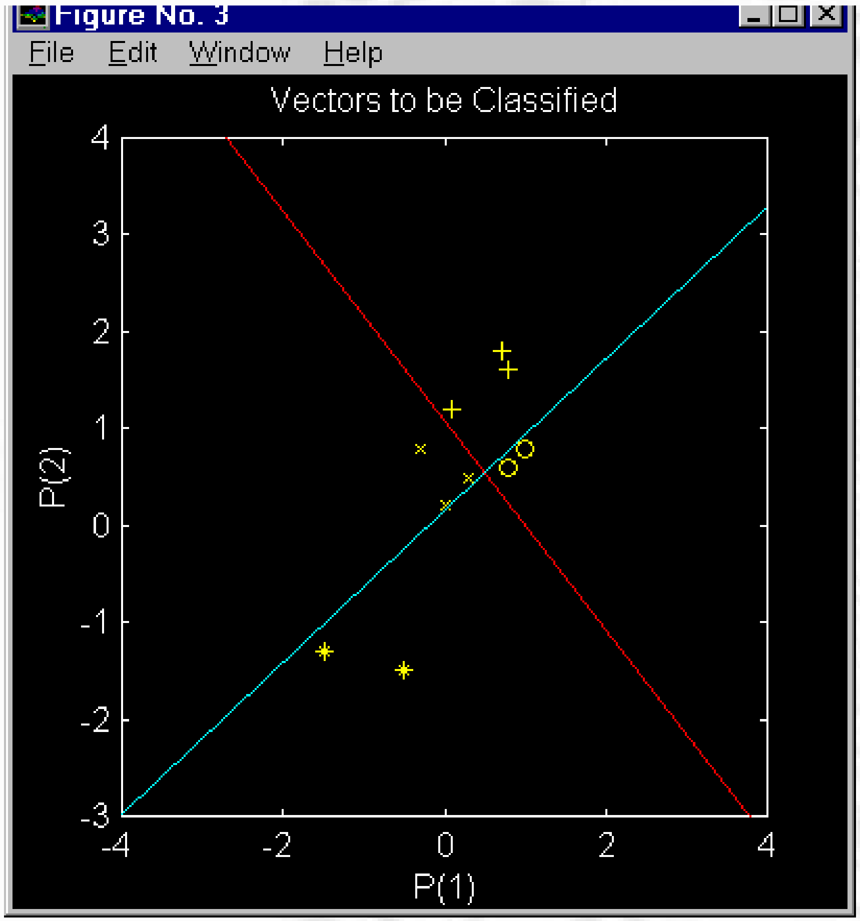




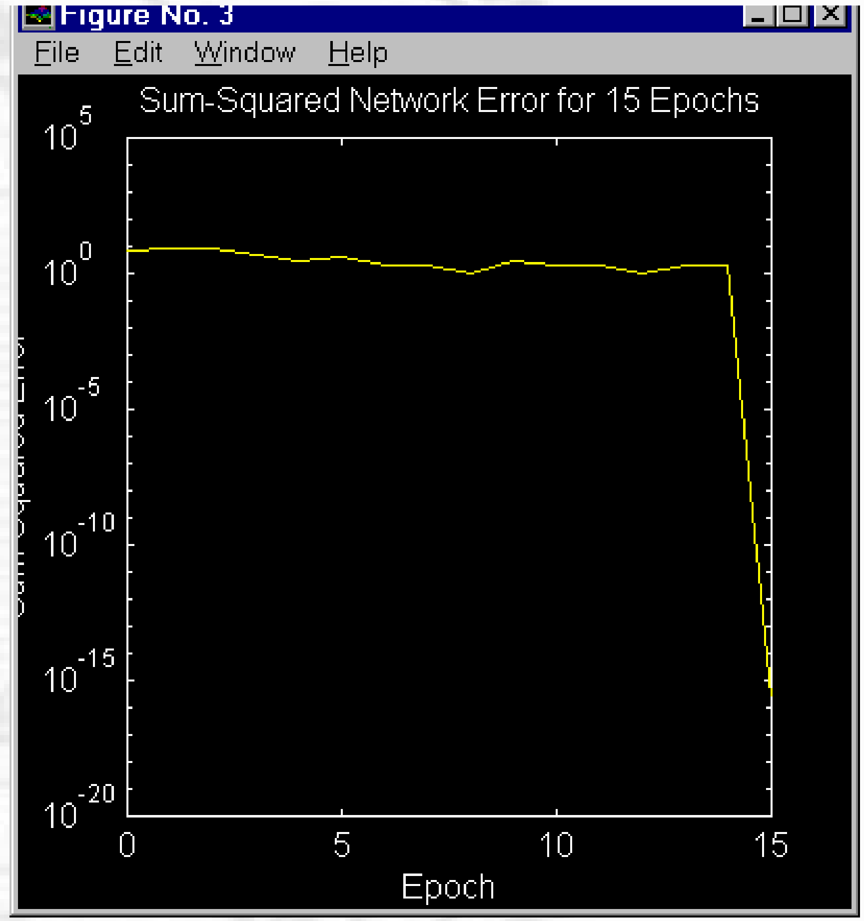
$$y_1 = g(w_{11}x_1 + w_{12}x_2 + b_1)$$

$$y_2 = g(w_{21}x_1 + w_{22}x_2 + b_2)$$

Final training results



Error vs. training epoch



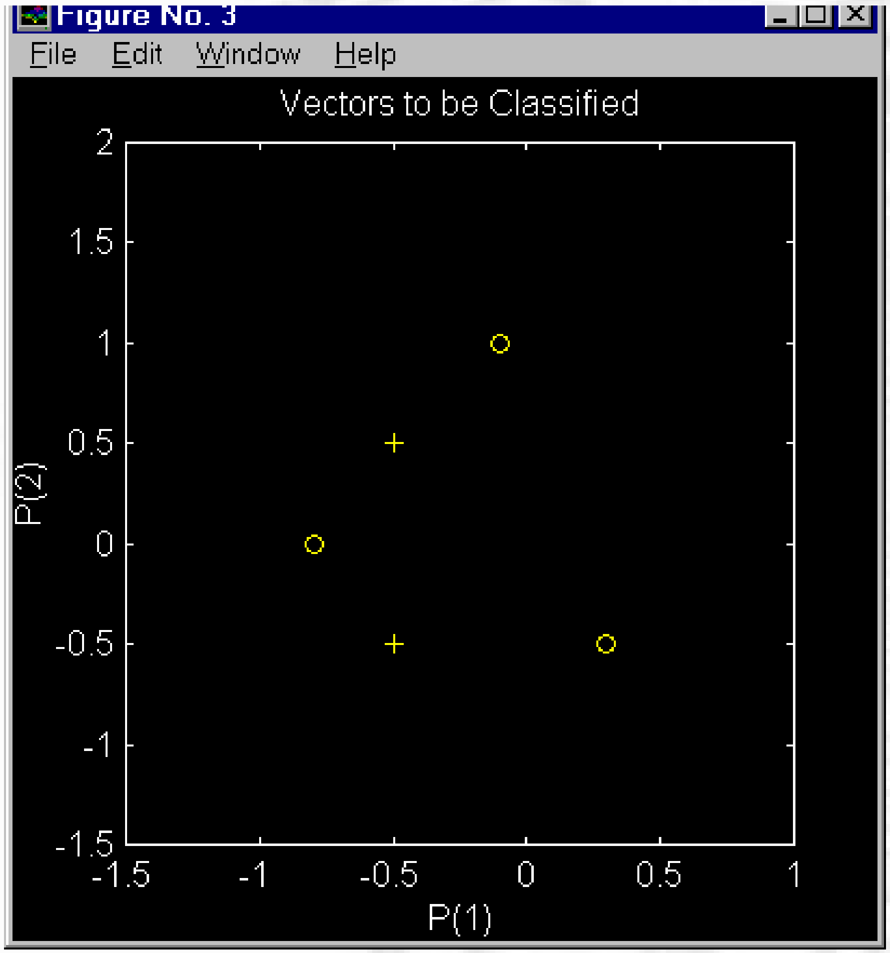


Figure No. 3
File Edit Window Help
Vectors to be Classified
P(2)
P(1)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5 -1 -0.5 0 0.5 1

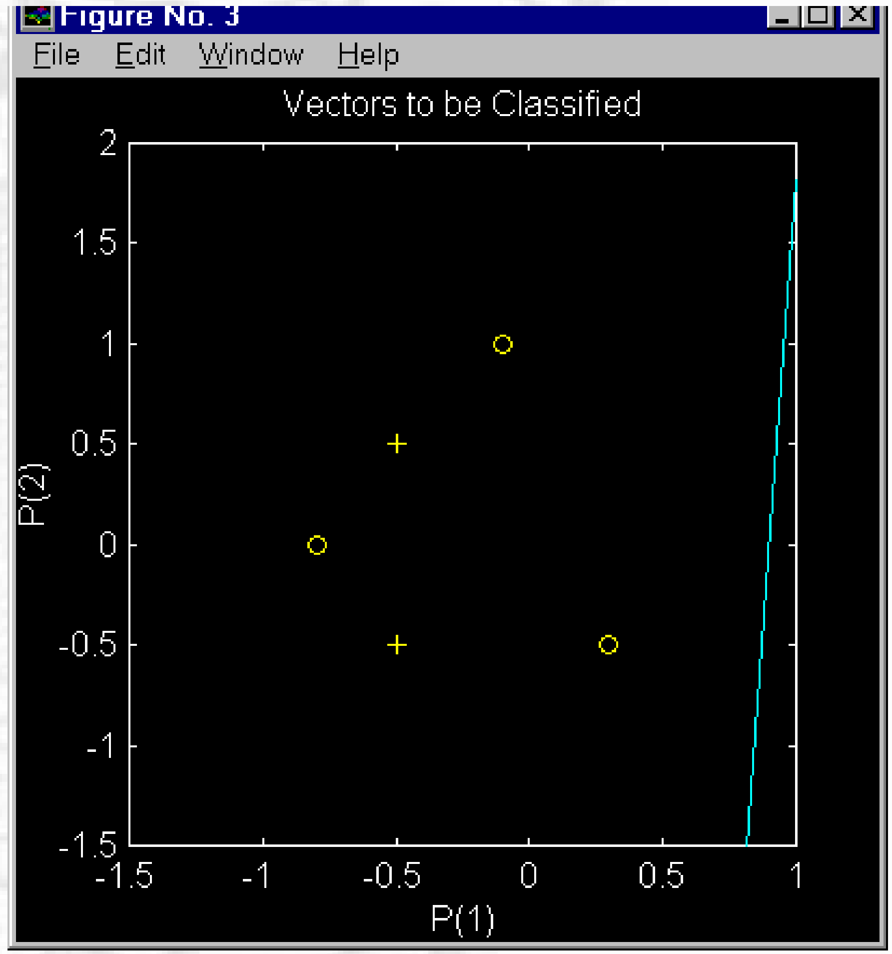
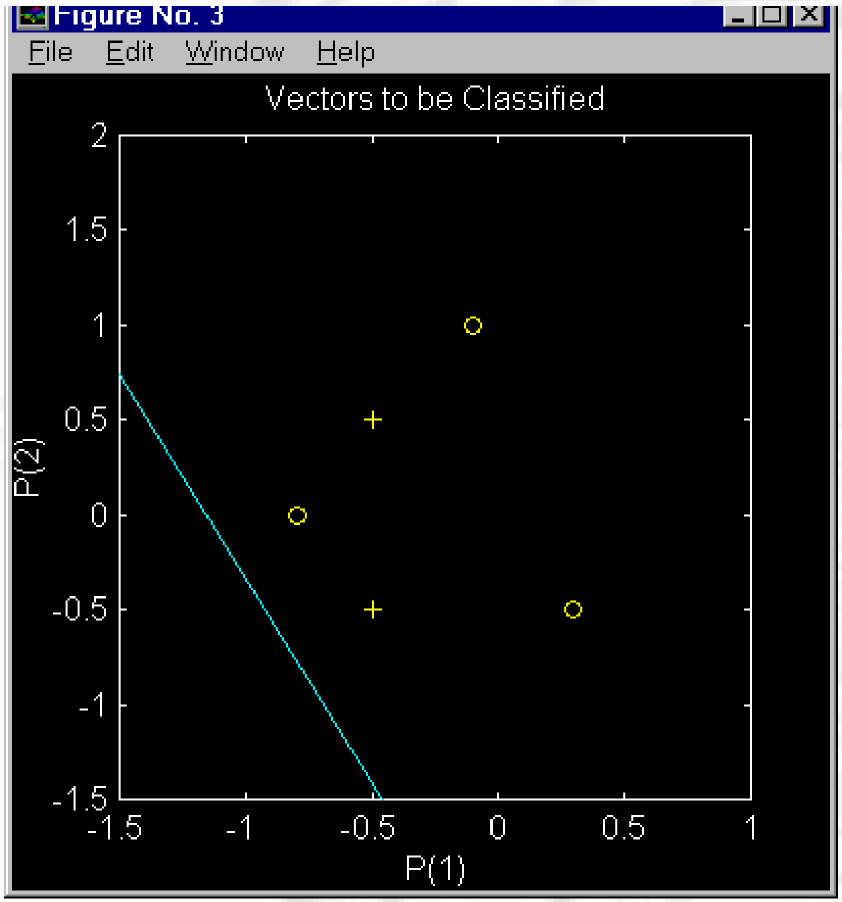


Figure No. 3
File Edit Window Help
Vectors to be Classified
P(2)
P(1)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5 -1 -0.5 0 0.5 1

Final training results



Error vs. training epoch